



CMPT 117
Computer Science
February 14, 2002
11.30am in class (Open Book)

Name _____

Student Number _____

NSID _____

Please read each question carefully before answering it. Answer each question in a separate booklet. All the code has been tested. State any assumptions you find necessary to make.

1. (5 marks)

Briefly describe what happens during compilation of a multi-file project. In particular, explain the role of the preprocessor, the compiler and the linker.

2. (2 marks)

If a variable is declared as a pointer, what must be stored in the variable?

3. (8 marks)

For the following declarations:

```
int *yaddr;  
long *lt;  
double *pt;  
int a;  
double c;
```

which of the following statements are valid? For each statement, indicate whether it is valid, and if it is not, state why or why not.

```
yaddr = &a;  
yaddr = &c;  
lt = a;  
pt = &c;
```

4. (10 marks)

What is the output of the following program?

```
#include <iostream>
using namespace std;
```

```
int main()
```

```
{
```

```
    int ia[ 10 ] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

```
    int j;
```

```
    int *ip = ia;
```

```
    for ( j = 1; j < 8; j += 2 )
```

```
        cout << *( ip + j ) << endl;
```

```
    for ( ; j > 0; j -= 2 )
```

```
        cout << ip[ j ] << endl;
```

```
    return 0;
```

```
}
```

5. Here is a struct definition very similar to one you received in one of your labs, along with a pointer definition:

```
struct Link {
    int data;
    Link *Next;
};
Link *Head = 0;
```

Assume that **Head** always points to the head of a possibly empty linked list structure.

6. (10 marks)

Write some code (it does not have to be a function) that removes only the *first* element of the list of Links pointed to by **Head**. Your code should do nothing if the list is empty.

If you wish, you can write a function **removeFirst(Head)** but this is more complicated and not required for this question.

7. (20 marks)

Write some code that removes the *last* element of a list of Links pointed to by **Head**.

6. (20 marks)

A **Rint** is a special kind of integer-like object. It is an integer that rounds *up* or *down*, rather than truncates when assigned, or initialized with, a floating point value. Normally, an integer (**int**) variable is rounded up or down using logic such as the following:

```
float f0, f1;
```

```
f0 = 2.6;
```

```
f1 = 5.2;
```

```
int i0, i1;
```

```
i0 = int( f0 + 0.5 ); //OR i0 = static_cast<int>( f0 + 0.5 );
```

```
i1 = int( f1 + 0.5 ); //OR i1 = static_cast<int>( f1 + 0.5 );
```

```
cout << i0 << endl << i1 << endl;
```

The above code will print the integer values 3 and 5.

The following class definition shows the part of the public interface of a **Rint** object.

```
class Rint {
```

```
public:
```

```
    Rint();
```

```
    Rint( float i );
```

```
    Rint( const Rint &);
```

```
    Rint &operator=( float & );
```

```
private:
```

```
    int _i;
```

```
};
```

Write implementations of the three constructors and the assignment operator given in the above definition, so that a **Rint** rounds when assigned or initialized with a float value. (Do not worry about other possibilities such as initializing with a **double** or a **long**). Indicate which of the constructors is the default constructor and which is the copy constructor.

85 marks total

—END OF EXAM—